CS 2100: Data Structures & Algorithms 1

Advanced Sorts (Part II)

Quicksort; Discussion on Hybrid Algorithms

Dr. Nada Basit // basit@virginia.edu

Spring 2022

Friendly Reminders

- The University updated the mask policy. As per my Request on Mar 28, 2022 (see Collab), I would greatly appreciate if you would do me a kind favor by **continuing to wear your masks** in CS 2100 (Ridley G008). I know it is a lot to ask, and it is **voluntary**, but I appreciate your understanding.
- If you forget your mask (or mask is lost/broken), I have a few available
  - Just come up to me at the start of class and ask!
- No eating or drinking in the classroom, please
- Our lectures will be **recorded** (see Collab) – please allow 24-48 hrs to post
- If you feel **unwell**, or think you are, please stay home
  - *We will work with you!*
  - At home: eye mask instead! Get some rest ☺

How Might YOU design an efficient Sorting Algorithm?

Think about…

How might you design a brand-new App to accomplish X?
How might you design a brand-new Streaming Service?
How might you design a brand-new Data Structure?
How might you design a brand-new SIS software?

Hybrid Sorts & Other Sorting Algorithms

Hybrid Sorts

- Some sorting algorithms (like Java's internal one) will look at **properties** of the list and **call different algorithms** depending on the situation.
- For example:
  - **Insertion sort** is faster than merge/quick on smaller lists
  - **Insertion sort** is faster on almost sorted lists

- Strategy:
  - **Switch to insertion sort once recursive calls get small** (small could be ~100-150 elements; or even down to 30-50 elements) or on an almost sorted list → **speedup!**
  - You could start with **quicksort** or **mergesort** which is **log-linear** time, and stop when the size of the list is small (e.g. 30-40) then switch to ***insertion sort*** (although **quadratic**, it is *faster on smaller lists!)* In the base case, check if size < **threshold** (instead of 1) if so, call ***insertion sort!***

Other Sorting Algorithms

- There are MANY more… but to name a few…

- **Heap Sort:** We haven't seen this data structure, so we will study this a little later
- **Radix Sort:** Uses values of digits to sort numbers very quickly.
- **TimSort:** What Java `Collections.sort()` uses
- ...and many others.

Fun and Colorful Quicksort Animation

https://github.com/jyahn/quicksort-visualization

15 Sorting Algorithms in 6 Minutes

Let’s watch this fun video that compares different sorts visually and audibly! ☺